

Package ‘spgrass6’

March 19, 2012

Version 0.7-10

Date 2012-03-18

Title Interface between GRASS 6+ geographical information system and R

Author Roger Bivand

Maintainer Roger Bivand <Roger.Bivand@nhh.no>

Description Interpreted interface between GRASS 6+ geographical information system and R, based on starting R from within the GRASS environment, or running free-standing R in a temporary GRASS location; the package provides facilities for using all GRASS commands from the R command line.

Depends R (>= 2.12), sp (>= 0.9), XML

Suggests rgdal (>= 0.6-7)

SystemRequirements GRASS (>= 6.3)

License GPL (>= 2)

URL <http://grass.osgeo.org/>

Collate AAA.R options.R spgrass6.R bin_link.R vect_link.R initGRASS.R xml1.R

Repository CRAN

Date/Publication 2012-03-19 11:12:45

R topics documented:

spgrass6-package	2
execGRASS	3
gmeta6	5
initGRASS	6
readRAST6	8
readVECT6	10

Index	14
--------------	-----------

Description

Interpreted interface between GRASS geographical information system, versions 6 and 7, and R, based on starting R from within the GRASS environment, or on running R stand-alone and creating a throw-away GRASS environment from within R. The interface uses classes defined in the sp package to hold spatial data.

Details

Index:

readRAST6	read GRASS raster files
writeRAST6	write GRASS raster files
readVECT6	read GRASS vector object files
writeVECT6	write GRASS vector object files
gmeta6	read GRASS metadata from the current LOCATION
getLocationProj	return a PROJ.4 string of projection information
gmeta2grd	create a GridTopology object from the GRASS region
vInfo	return vector geometry information
vColumns	return vector database columns information
vDataCount	return count of vector database rows
vect2neigh	return area neighbours with shared boundary length

Further information may be found in the document doc/spgrass_0.3.pdf in the directory returned by `system.file("", package="spgrass6")`.

Author(s)

Roger Bivand

Maintainer: Roger Bivand <Roger.Bivand@nhh.no>

Examples

```
if (nchar(Sys.getenv("GISRC")) > 0) {
  require(rgdal)
  soilsph <- readRAST6("soils.ph", ignore.stderr=TRUE, plugin=FALSE)
  summary(soilsph)
  grd <- gmeta2grd(ignore.stderr=TRUE)
  grd
  set.seed(1)
  smple <- overlay(soilsph, spsample(soilsph, 200, "random"))
  summary(smple)
  writeVECT6(smple, "sp_dem", v.in.ogr_flags="overwrite", ignore.stderr=TRUE)
  bugsDF <- readVECT6("bugsites", ignore.stderr=TRUE, mapset="PERMANENT")
  summary(bugsDF)
```

```

vInfo("streams", ignore.stderr=TRUE)
vColumns("streams", ignore.stderr=TRUE)
vDataCount("streams", ignore.stderr=TRUE)
streams <- readVECT6("streams", type="line,boundary", remove.duplicates=FALSE, ignore.stderr=TRUE, plugin=FALSE)
summary(streams)
}

```

execGRASS

Run GRASS commands

Description

The functions provide an interface to GRASS commands run through system, based on the values returned by the `--interface` description flag using XML parsing. If required parameters are omitted, and have declared defaults, the defaults will be used.

Usage

```

execGRASS(cmd, flags = NULL, ..., parameters = NULL, intern = FALSE,
  ignore.stderr = NULL, Sys_ignore.stdout=FALSE, Sys_wait=TRUE,
  Sys_input=NULL, Sys_show.output.on.console=TRUE, Sys_minimized=FALSE,
  Sys_invisible=TRUE, echoCmd=NULL)
doGRASS(cmd, flags = NULL, ..., parameters = NULL, echoCmd=NULL)
parseGRASS(cmd)
## S3 method for class 'GRASS_interface_desc'
print(x, ...)
getXMLencoding()
setXMLencoding(enc)

```

Arguments

<code>cmd</code>	GRASS command name
<code>flags</code>	character vector of GRASS command flags
<code>...</code>	for <code>execGRASS</code> and <code>doGRASS</code> , GRASS module parameters given as R named arguments directly. For the <code>print</code> method, other arguments to <code>print</code> method
<code>parameters</code>	list of GRASS command parameters, used if GRASS parameters are not given as R arguments directly; the two methods for passing GRASS parameters may not be mixed
<code>intern</code>	a logical (not 'NA') which indicates whether to make the output of the command an R object. Not available unless 'popen' is supported on the platform
<code>ignore.stderr</code>	default NULL, taking the value set by <code>set.ignore.stderrOption</code> , a logical indicating whether error messages written to 'stderr' should be ignored
<code>Sys_ignore.stdout</code> , <code>Sys_wait</code> , <code>Sys_input</code>	pass extra arguments to system
<code>Sys_show.output.on.console</code> , <code>Sys_minimized</code> , <code>Sys_invisible</code>	pass extra arguments to system on Windows systems only

echoCmd	default NULL, taking the logical value set by <code>set.echoCmdOption</code> , print GRASS command to be executed to console
x	object to be printed
enc	character string to replace UTF-8 in header of XML data generated by GRASS module <code>-interface-description</code> output when the internationalised messages are not in UTF-8 (known to apply to French, which is in latin1)

Details

`parseGRASS` checks to see whether the GRASS command has been parsed already and cached in this session; if not, it reads the interface description, parses it and caches it for future use. `doGRASS` assembles a proposed GRASS command with flags and parameters as a string, wrapping `parseGRASS`, and `execGRASS` is a wrapper for `doGRASS`, running the command through `system` (from 0.7-4, the `...` argument is not used for passing extra arguments for `system`). The command string is termed `proposed`, because not all of the particular needs of commands are provided by the interface description, and no check is made for the existence of input objects. Support for multiple parameter values added with help from Patrick Caldon. Support for defaults and for direct use of GRASS parameters instead of a parameter list suggested by Rainer Krug.

Value

`parseGRASS` returns a `GRASS_interface_desc` object, `doGRASS` returns a character string with a proposed GRASS command, and `execGRASS` returns what `system` returns, particularly depending on the `intern` argument.

Note

If any package command fails with a UTF-8 error from the XML package, try using `setXMLencoding` to work around the problem that GRASS modules declare `-interface-description` output as UTF-8 without ensuring that it is (French is of 6.4.0 RC5 latin1).

Author(s)

Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no>

See Also

[system](#)

Examples

```
if (nchar(Sys.getenv("GISRC")) > 0) {
  oechoCmd <- get.echoCmdOption()
  set.echoCmdOption(TRUE)
  print(parseGRASS("r.slope.aspect"))
  doGRASS("r.slope.aspect", flags=c("overwrite"),
    elevation="elevation.dem", slope="slope", aspect="aspect")
  pars <- list(elevation="elevation.dem", slope="slope", aspect="aspect")
  doGRASS("r.slope.aspect", flags=c("overwrite"), parameters=pars)
  print(parseGRASS("r.buffer"))
}
```

```

doGRASS("r.buffer", flags=c("overwrite"), input="bugsites", output="bmap",
  distances=seq(1000,15000,1000))
pars <- list(input="bugsites", output="bmap", distances=seq(1000,15000,1000))
doGRASS("r.buffer", flags=c("overwrite"), parameters=pars)
set.echoCmdOption(oechoCmd)
}

```

gmeta6

*Reads GRASS metadata from the current LOCATION***Description**

GRASS LOCATION metadata are read into a list in R; helper function getLocationProj returns an sproj-compliant PROJ.4 string of projection information. The helper function gmeta2grd creates a GridTopology object from the current GRASS mapset region definitions.

Usage

```

gmeta6(ignore.stderr = FALSE)
getLocationProj(ignore.stderr = FALSE)
gmeta2grd(ignore.stderr = FALSE)
## S3 method for class 'gmeta6'
print(x, ...)
get.ignore.stderrOption()
get.stop_on_no_flags_parasOption()
get.useGDALOption()
get.pluginOption()
get.echoCmdOption()
set.ignore.stderrOption(value)
set.stop_on_no_flags_parasOption(value)
set.useGDALOption(value)
set.pluginOption(value)
set.echoCmdOption(value)

```

Arguments

ignore.stderr	default FALSE, can be set to TRUE to silence system() output to standard error; does not apply on Windows platforms
x	S3 object returned by gmeta6
...	arguments passed through print method
value	logical value for setting options on ignore.stderr set by default on package load to FALSE, stop_on_no_flags_paras set by default on package load to TRUE, useGDAL set by default on package load to TRUE, plugin set by default on package load to NULL, echoCmd set by default on package load to FALSE

Value

Returns list of g.gisenv, g.region -g3, and g.proj values

Author(s)

Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no.>

Examples

```
if (nchar(Sys.getenv("GISRC")) > 0) {
  G <- gmeta6()
  print(G)
  CRS(getLocationProj())
  grd <- gmeta2grd()
  print(grd)
  ncells <- prod(slot(grd, "cells.dim"))
  df <- data.frame(k=rep(1, ncells))
  mask_SG <- SpatialGridDataFrame(grd, data=df)
  print(summary(mask_SG))
}
```

initGRASS

Initiate GRASS session

Description

Run GRASS interface in an R session not started within GRASS. In general, most users will use `initGRASS` in throwaway locations, to use GRASS modules on R objects without the need to define and populate a location. The function initializes environment variables used by GRASS, the `.gisrc` used by GRASS for further environment variables, and a temporary location.

The locking functions are used internally, but are exposed for experienced R/GRASS scripters needing to use the GRASS module “`g.mapset`” through `initGRASS` in an existing GRASS location. In particular, “`g.mapset`” may leave a `.gislock` file in the current MAPSET, so it may be important to call `unlink_.gislock` to clean up before quitting the R session.

Usage

```
initGRASS(gisBase, home, SG, gisDbase, location, mapset, override = FALSE, use_g.dirseps.exe = TRUE, pi
get.GIS_LOCK()
set.GIS_LOCK(pid)
unset.GIS_LOCK()
unlink_.gislock()
```

Arguments

<code>gisBase</code>	The directory path to GRASS binaries and libraries
<code>home</code>	The directory in which to create the <code>.gisrc</code> file; defaults to <code>\$HOME</code> on Unix systems and to <code>USERPROFILE</code> on Windows systems; can usually be set to <code>tempdir()</code>
<code>SG</code>	An optional <code>SpatialGrid</code> object to define the <code>DEFAULT_WIND</code> of the temporary location

gisDbase	if missing, tempdir() will be used; GRASS GISDBASE directory for the working session
location	if missing, basename(tempfile()) will be used; GRASS location directory for the working session
mapset	if missing, basename(tempfile()) will be used; GRASS mapset directory for the working session
override	default FALSE, set to TRUE if accidental trashing of GRASS .gisrc files and locations is not a problem
use_g.dirseps.exe	default TRUE; when TRUE appears to work for WinGRASS Native binaries, when FALSE for QGIS GRASS binaries; ignored on other platforms.
pid	default as.integer(round(runif(1, 1, 1000))), integer used to identify GIS_LOCK; the value here is arbitrary, but probably should be set correctly

Details

The function establishes an out-of-GRASS working environment providing GRASS commands with the environment variable support required, and may also provide a temporary location for use until the end of the running R session if the home argument is set to tempdir(), and the gisDbase argument is not given. Running gmeta6 shows where the location is, should it be desired to archive it before leaving R.

Value

The function runs gmeta6 before returning the current values of the running GRASS session that it provides.

Note

If any package command fails with a UTF-8 error from the XML package, try using setXMLencoding to work around the problem that GRASS modules declare –interface-description output as UTF-8 without ensuring that it is (French is of 6.4.0 RC5 latin1).

Author(s)

Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no>

See Also

[gmeta6](#)

Examples

```
## Not run:
initGRASS("/usr/local/grass-6.4.0", home=tempdir())
initGRASS("C:/GRASS", home=tempdir())

## End(Not run)
```

readRAST6 *Read and write GRASS 6+ raster files*

Description

Read GRASS 6+ raster files from GRASS 6+ into R SpatialGridDataFrame objects, and write single columns of R SpatialGridDataFrame objects to GRASS 6+. readRAST6 and writeRAST6 use temporary binary files and r.out.bin and r.in.bin rather than the temporary ASCII files used in earlier implementations. The earlier versions may still be used in a transition period.

Usage

```
readRAST6(vname, cat=NULL, ignore.stderr = NULL, NODATA=NULL, plugin=NULL, mapset=NULL, useGDAL=NULL,
writeRAST6(x, vname, zcol = 1, NODATA=NULL, ignore.stderr = NULL, useGDAL=NULL, overwrite=FALSE, flags=
```

Arguments

vname	A vector of GRASS 6.0 raster file names
cat	default NULL; if not NULL, must be a logical vector matching vname, stating which (CELL) rasters to return as factor
ignore.stderr	default NULL, taking the value set by set.ignore.stderrOption, can be set to TRUE to silence system() output to standard error; does not apply on Windows platforms
plugin	default NULL does auto-detection, changes to FALSE if vname is longer than 1, and a sanity check will be run on raster and current region, and the function will revert to FALSE if mismatch is found; if TRUE, the plugin is available and the raster should be read in its original region and resolution; if the plugin is used, no further arguments other than mapset are respected
mapset	default NULL, if plugin is TRUE, the mapset of the file to be imported will be autodetected; if not NULL and if plugin is TRUE, a character string overriding the autodetected mapset, otherwise ignored
useGDAL	default NULL, taking the value set by set.useGDALOption; use r.out.gdal or plugin and readGDAL if autodetected or plugin=TRUE; or for writing writeGDAL, GTiff, and r.in.gdal, if FALSE using r.out.bin or r.in.bin
close_OK	default TRUE - clean up possible open connections used for reading metadata; may be set to FALSE to avoid the side-effect of other user-opened connections being broken
drivername	default "GTiff"; a valid GDAL writable driver name to define the file format for intermediate files
driverFileExt	default NULL; otherwise string value of required driver file name extension
return_SGDF	default TRUE returning a SpatialGridDataFrame object, if FALSE, return a list with a GridTopology object, a list of bands, and a proj4string; see example below
x	A SpatialGridDataFrame object for export to GRASS as a raster layer

zcol	Attribute column number or name
NODATA	by default NULL, in which case it is set to one less than floor() of the data values, otherwise an integer NODATA value (required to be integer by GRASS r.out.bin)
overwrite	default FALSE, if TRUE inserts "overwrite" into the value of the flags argument if not already there to allow existing GRASS rasters to be overwritten
flags	default NULL, character vector, for example "overwrite"

Value

readRAST6 returns a SpatialGridDataFrame objects with an data.frame in the data slots, and with the projection argument set. Note that the projection argument set is the the GRASS rendering of proj4, and will differ from the WKT/ESRI rendering returned by readVECT6 in form but not meaning. They are exchangeable but not textually identical, usually with the +ellps= term replaced by ellipsoid parameters verbatim. If return_SGDF is FALSE, a list with a GridTopology object, a list of bands, and a proj4string is returned, with an S3 class attribute of "gridList".

Author(s)

Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no>

Examples

```
if (nchar(Sys.getenv("GISRC")) > 0) {
  require(rgdal)
  ois <- get.ignore.stderrOption()
  set.ignore.stderrOption(TRUE)
  get.useGDALOption()
  spear <- readRAST6(c("geology", "elevation.dem"), cat=c(TRUE, FALSE),
    useGDAL=FALSE)
  spear <- readRAST6(c("geology", "elevation.dem"), cat=c(TRUE, FALSE),
    useGDAL=TRUE)
  print(table(spear$geology))
  execGRASS("r.stats", flags=c("c", "l", "quiet"), input="geology")
  boxplot(spear$elevation.dem ~ spear$geology)
  spear$sqdem <- sqrt(spear$elevation.dem)
  if ("GRASS" %in% gdalDrivers()$name) {
    execGRASS("g.region", rast="elevation.dem")
    dem1 <- readRAST6("elevation.dem", plugin=TRUE, mapset="PERMANENT")
    print(summary(dem1))
    execGRASS("g.region", rast="elevation.dem")
  }
  writeRAST6(spear, "sqdemSP", zcol="sqdem")
  execGRASS("r.info", map="sqdemSP")
  execGRASS("g.remove", rast="sqdemSP")
  writeRAST6(spear, "sqdemSP", zcol="sqdem", useGDAL=TRUE)
  execGRASS("r.info", map="sqdemSP")
  print(system.time(sqdemSP <- readRAST6(c("sqdemSP", "elevation.dem"),
    useGDAL=TRUE, return_SGDF=FALSE)))
  print(system.time(sqdemSP <- readRAST6(c("sqdemSP", "elevation.dem"),
    useGDAL=TRUE, return_SGDF=TRUE)))
}
```

```

print(system.time(sqdemSP <- readRAST6(c("sqdemSP", "elevation.dem"),
  useGDAL=FALSE, return_SGDF=TRUE)))
print(system.time(sqdemSP <- readRAST6(c("sqdemSP", "elevation.dem"),
  useGDAL=FALSE, return_SGDF=FALSE)))
str(sqdemSP)
mat <- do.call("cbind", sqdemSP$dataList)
str(mat)
print(system.time(SGDF <- SpatialGridDataFrame(grid=sqdemSP$grid,
  proj4string=sqdemSP$proj4string, data=as.data.frame(sqdemSP$dataList))))
summary(SGDF)
execGRASS("g.remove", rast="sqdemSP")
if (execGRASS("g.version", intern=TRUE) > "GRASS 7") {
  execGRASS("r.mapcalc", expression="quads0 = quads - 1")
} else {
  execGRASS("r.mapcalculator", outfile="quads0",
    amap="quads", formula='A - 1')
}
execGRASS("r.stats", flags="c", input="quads0")
quads0 <- readRAST6("quads0")
print(table(quads0$quads0))
quads0 <- readRAST6("quads0", plugin=FALSE)
print(table(quads0$quads0))
execGRASS("g.remove", rast="quads0")
set.ignore.stderrOption(ois)
}

```

readVECT6

Read and write GRASS 6+ vector object files

Description

readVECT6 moves one GRASS 6+ vector object file with attribute data through a temporary shapefile to a Spatial*DataFrame object of type determined by the GRASS 6+ vector object; writeVECT6 moves a Spatial*DataFrame object through a temporary shapefile to a GRASS vector object file. vect2neigh returns neighbour pairs with shared boundary length as described by Markus Neteler, in <https://stat.ethz.ch/pipermail/r-sig-geo/2005-October/000616.html>. cygwin_clean_temp can be called to try to clean the GRASS mapset-specific temporary directory under cygwin.

Usage

```

readVECT6(vname, layer, type=NULL, plugin=NULL, remove.duplicates = TRUE,
  ignore.stderr = NULL, with_prj=TRUE, with_c=FALSE, mapset=NULL,
  pointDropZ=FALSE, driver="ESRI Shapefile")
writeVECT6(SDF, vname, v.in.ogr_flags=NULL, ignore.stderr = NULL,
  driver="ESRI Shapefile")
vInfo(vname, layer, ignore.stderr = NULL)
vColumns(vname, layer, ignore.stderr = NULL)
vDataCount(vname, layer, ignore.stderr = NULL)
vect2neigh(vname, ID=NULL, ignore.stderr = NULL, remove=TRUE, vname2=NULL,

```

```
units="k")
```

Arguments

vname	A GRASS 6+ vector file name
layer	a layer name (integer in GRASS 6, string in GRASS 7); if missing set to default of 1
type	override type detection when multiple types are non-zero, passed to v.out.ogr
plugin	default NULL for auto-detection, may be set to FALSE to avoid or TRUE if the plugin is known to be available; if the plugin is used, no further arguments other than mapset are respected
remove.duplicates	In line and area vector objects, multiple geometrical features may be associated with a single cat number, leading to duplication of data rows; this argument attempts to combine the geometrical features so that they match a single data row
ignore.stderr	default NULL, taking the value set by set.ignore.stderrOption, can be set to TRUE to silence system() output to standard error; does not apply on Windows platforms
with_prj	default TRUE, write ESRI-style PRJ file for transferred data
with_c	if TRUE, export features with category (labeled) only; by default all features are exported
mapset	if plugin is TRUE, the mapset of the file to be imported may be changed from the current mapset by passing a character string
pointDropZ	default FALSE, if TRUE, discard third coordinates for point geometries; third coordinates are always discarded for line and polygon geometries
driver	default "ESRI Shapefile"; a valid OGR writable driver name to define the file format for intermediate files, one of c("GML", "SQLite"), c("ESRI_Shapefile", "MapInfo_File")
SDF	A Spatial*DataFrame to be moved to GRASS6 as a vector object, for SpatialPointsDataFrame, SpatialLinesDataFrame, and SpatialPolygonsDataFrame objects
v.in.ogr_flags	Character vector containing additional optional flags and/or options for v.in.ogr, particularly "o" and "overwrite"
ID	A valid DB column name for unique identifiers (optional)
remove	default TRUE, remove copied vectors created in vect2neigh
vname2	If on a previous run, remove was FALSE, the name of the temporary vector may be given to circumvent its generation
units	default "k"; see GRASS v. to.db manual page for alternatives

Value

readVECT6 imports a GRASS6+ vector object into a Spatial*DataFrame object with the type determined by the type of the GRASS6 vector object; getSites6 returns a data frame. vect2neigh returns a data frame object with left and right neighbours and boundary lengths, also given class GRASSneigh and spatial.neighbour (as used in spdep). The incantation to retrieve the neighbours list is sn2listw(vect2neigh())\$neighbours, and to retrieve the boundary lengths: sn2listw(vect2neigh())\$weights. The GRASSneigh object has two other useful attributes: external is a vector giving the length of shared boundary between each polygon and the external area, and total giving each polygon's total boundary length.

Note that GRASS6 vectors should be rebuilt by running "v.build.all" in each mapset.

Note

Please note that the OGR drivers used may not handle missing data gracefully. From rgdal release 0.5-27, missing values are taken as unset OGR field values. If the OGR driver encodes them in this way, NAs will be moved across the interface correctly from R to GRASS, and from GRASS to R using the OGR GRASS vector plugin. Work is continuing to correct v.out.ogr so that it emits unset fields, which affects users with no OGR GRASS plugin for the present. Thanks to Dylan Beaudette for helping with missing data handling.

Please also note that, on Windows and Cygwin systems, the temporary shapefiles are not removed by the interface functions, nor can GRASS remove them on termination - they must for the time being be removed manually. Windows believes that the GDAL/OGR library is still using them.

Author(s)

Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no.>

Examples

```
if (nchar(Sys.getenv("GISRC")) > 0) {
  require(rgdal)
  ois <- get.ignore.stderrOption()
  set.ignore.stderrOption(TRUE)
  if (execGRASS("g.version", intern=TRUE) > "GRASS 7") {
    execGRASS("v.info", map="bugsites", layer="1")
  } else {
    execGRASS("v.info", map="bugsites", layer=1L)
  }
  print(vInfo("bugsites"))
  bugs <- readVECT6("bugsites", plugin=NULL)
  print(summary(bugs))
  bugs1 <- readVECT6("bugsites", plugin=FALSE)
  print(summary(bugs1))
  writeVECT6(bugs, "newbugs", v.in.ogr_flags=c("o", "overwrite"))
  if (execGRASS("g.version", intern=TRUE) > "GRASS 7") {
    execGRASS("v.info", map="newbugs", layer="1")
  } else {
    execGRASS("v.info", map="newbugs", layer=1L)
  }
  nbugs <- readVECT6("newbugs")
}
```

```
print(summary(nbugs))
print(vInfo("roads"))
roads <- readVECT6("roads")
print(summary(roads))
set.ignore.stderrOption(ois)
}
```

Index

- *Topic **package**
 - spgrass6-package, 2
- *Topic **spatial**
 - execGRASS, 3
 - gmeta6, 5
 - initGRASS, 6
 - readRAST6, 8
 - readVECT6, 10
 - spgrass6-package, 2
- doGRASS (execGRASS), 3
- execGRASS, 3
- get.echoCmdOption (gmeta6), 5
- get.GIS_LOCK (initGRASS), 6
- get.ignore.stderrOption (gmeta6), 5
- get.pluginOption (gmeta6), 5
- get.stop_on_no_flags_parasOption (gmeta6), 5
- get.useGDALOption (gmeta6), 5
- getLocationProj (gmeta6), 5
- getXMLencoding (execGRASS), 3
- gmeta2grd (gmeta6), 5
- gmeta6, 5, 7
- initGRASS, 6
- parseGRASS (execGRASS), 3
- print.gmeta6 (gmeta6), 5
- print.GRASS_interface_desc (execGRASS), 3
- readRAST6, 8
- readVECT6, 10
- set.echoCmdOption (gmeta6), 5
- set.GIS_LOCK (initGRASS), 6
- set.ignore.stderrOption (gmeta6), 5
- set.pluginOption (gmeta6), 5
- set.stop_on_no_flags_parasOption (gmeta6), 5
- set.useGDALOption (gmeta6), 5
- setXMLencoding (execGRASS), 3
- spgrass6 (spgrass6-package), 2
- spgrass6-package, 2
- system, 4
- unlink_.gislock (initGRASS), 6
- unset.GIS_LOCK (initGRASS), 6
- vColumns (readVECT6), 10
- vDataCount (readVECT6), 10
- vect2neigh (readVECT6), 10
- vInfo (readVECT6), 10
- writeRAST6 (readRAST6), 8
- writeVECT6 (readVECT6), 10